MEMORANDUM
RM-3084-PR
APRIL 1962

POLYNOMIAL APPROXIMATION– A NEW COMPUTATIONAL TECHNIQUE IN DYNAMIC PROGRAMMING–I. ALLOCATION PROCESSES

Richard Bellman, Robert Kalaba and Bella Kotkin

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-3084-PR
APRIL 1962

POLYNOMIAL APPROXIMATION– A NEW COMPUTATIONAL TECHNIQUE IN DYNAMIC PROGRAMMING–I: ALLOCATION PROCESSES

Richard Bellman, Robert Kalaba and Bella Kotkin

This research is sponsored by the United States Air Force under Project RAND – Contract No. AF 49(638)-700 – monitored by the Directorate of Development Planning, Deputy Chief of Staff, Research and Technology, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force. Permission to quote from or reproduce portions of this Memorandum must be obtained from The RAND Corporation.

The RAND Corporation
1700 MAIN ST • SANTA MONICA • CALIFORNIA

PREFACE

Part of the Project RAND research program consists of basic supporting studies in mathematics. One aspect of this is concerned with optimization processes. In this field, the technique of dynamic programming has developed into a powerful mathematical tool.

In the present Memorandum the authors investigate the applicability of polynomial approximation as an adjunct to the technique of dynamic programming.

SUMMARY

In this Memorandum the authors initiate the application of the simple yet powerful computational technique of polynomial approximation to problems in dynamic programming.

The theoretical applicability of orthogonal polynomials is first discussed and then applied to one— and two—dimensional allocation problems. Numerical results obtained from FORTRAN programs involving Legendre polynomials are presented.

CONTENTS

POLYNOMIAL APPROXIMATION—A NEW COMPUTATIONAL TECHNIQUE IN DYNAMIC PROGRAMMING—I: ALLOCATION PROCESSES

1. INTRODUCTION

The problem of maximizing the function

$$(1.1) \qquad F(x_1, x_2, \ldots, x_N) = g_1(x_1) + g_2(x_2) + \cdots + g_N(x_N)$$

over the domain

$$(1.2) \qquad x_1 + x_2 + \cdots + x_N = x, \quad x_i \geq 0$$

can be reduced via familiar dynamic programming techniques (see [1]) to that of determining the sequence of functions $\{f_n(x)\}$ generated by means of the recurrence relation

$$(1.3) \qquad f_1(x) = g_1(x),$$

$$f_{n+1}(x) = \max_{0 \leq y \leq x} [g_{n+1}(y) + f_n(x - y)].$$

The problem can thus be solved numerically in a very simple fashion, regardless of the complexity of the functions $g_1(x)$. A number of important allocation processes can be resolved in this way. If we consider cases in which two distinct types of resources must be allocated, we face the problem of maximizing the function

$$(1.4) \qquad F(x_1, x_2, \ldots, x_N; y_1, y_2, \ldots, y_N)$$

$$= g_1(x_1, y_1) + g_2(x_2, y_2) + \cdots + g_N(x_N, y_N)$$

over the domain

$$(1.5) \qquad x_1 + x_2 + \cdots + x_N = x, \quad x_i \geq 0,$$

$$y_1 + y_2 + \cdots + y_N = y, \quad y_i \geq 0.$$

Theoretically, there is no difficulty in applying the same techniques.

The maximization problem can be reduced to the determination of the sequence $\{f_n(x,y)\}$ generated by means of the recurrence relation

$$(1.6) \qquad f_{n+1}(x,y) = \max_{\substack{0 \leq w \leq x \\ 0 \leq r \leq y}} [g_{n+1}(w,r) + f_n(x - w, y - r)].$$

In principle, this equation can be solved computationally using the same technique that applies so well to (1.3). In practice (see [1] for a discussion), questions of time and accuracy arise. There are a number of ways of circumventing these difficulties, among which the Lagrange multiplier plays a significant role.

In this series of papers, we wish to present a number of applications of a new, simple and quite powerful method, that of polynomial approximation. We shall begin with a discussion of the allocation process posed in the foregoing paragraphs and continue, in subsequent papers, with a treatment of realistic trajectory and guidance processes. In a separate series of papers we shall apply this fundamental attack upon

dimensionality to the solution of a number of the equations of mathematical physics.

We would like to express our appreciation to Oliver Gross for the analytic solution of some test problems we used to check the accuracy of our techniques, and for his general interest in the program.

2. POLYNOMIAL APPROXIMATION

In the systematic study of dynamic programming as a computational algorithm given in [1], a function $f(x)$ is considered to be a table of values at an appropriate set of grid points:

(2.1)

x	$f(x)$
0	f_0
Δ	f_1
2Δ	f_2
$\vdots$	$\vdots$
$K\Delta$	f_K

In this way, the function $f(x)$ is stored in the computer. For functions of one variable with $K = 100$ or 1000, this is a reasonably convenient way to proceed. For functions of two variables, this procedure becomes a bit inconvenient since $(K + 1)$ values for x combined with $(K + 1)$ values for y yields a total set of $(K + 1)^2$ values. Consequently, when we encounter functions of three or more variables, we must

balance accuracy against time and the limited storage of contemporary computers in our choice of K.

The storage of functional values by means of a table of values is ideally suited to the treatment of problems involving functions of quite arbitrary form. It is, on the other hand, quite wasteful and inefficient if we are dealing with functions possessing a definite structure, which is to say, situations in which there is a high correlation between the values of $f(r\Delta)$ and $f(s\Delta)$. Since functions of this type occur in many important applications, and throughout mathematical physics, it is worthwhile to develop methods which take advantage of the "smoothness" of the function.

One such method is polynomial approximation, or to be precise, generalized polynomial approximation. We represent the function in the form

$$f(x) \simeq \sum_{k=1}^{M} a_k \varphi_k(x), \tag{2.2}$$

where the $\varphi_k(x)$ are known elementary functions such as x^k, $\sin kx$, $P_k(x)$ (the Legendre polynomial of degree k) or $T_k(x)$ (the Cebycev polynomial of degree k), and then store the function for all values of interest by means of the set of M coefficients $[a_1, a_2, \ldots, a_M]$.

It is important to point out that (2.2) is particularly useful in automatically furnishing the interpolation-values frequently needed in dynamic programming calculations. If

one uses a table of values of the form shown in (2.1), interpolation is frequently a source of difficulty.

To determine the coefficients a_k it is convenient to make the $\varphi_k(x)$ be an orthonormal set. Then

$$(2.3) \qquad a_k = \int_0^1 f(x)\varphi_k(x)dx,$$

assuming for the purposes of convenience that $0 \leq x \leq 1$. We can, of course, use a Cebycev fit instead, and we will explore this in subsequent papers. A priori, we would suspect that it would be more efficient to use a mean-square fit (implied by (2.3)), and take M to be larger, than to go to the trouble of determining the a_k to minimize the function

$$(2.4) \qquad \max_{0\leq x\leq 1} \left| f(x) - \sum_{k=1}^{M} a_k\varphi_k(x) \right|,$$

which for a fixed M may be expected to yield a more accurate approximation. Alternatively, one could use an approximation by polygonal functions [1].

To evaluate the a_k without requiring a knowledge of too many values of $f(x)$, we use a quadrature technique

$$(2.5) \qquad a_k \cong \sum_{i=1}^{R} w_i f(x_i)\varphi_k(\bar{x}_i),$$

where the weights w_i and the quadrature points $\bar{x}_i$ are chosen so that the equation

$$(2.6) \qquad \int_0^1 g(x)dx = \sum_{i=1}^{R} w_i g(\bar{x}_i)$$

is exact for polynomials in x of degree $(2R - 1)$ or less. This requirement narrows us down to the Gauss quadrature technique [2]. If we use generalized polynomials (expressions such as (2.6)), we will obtain different weights and quadrature points.

We may then store the function $f(x)$ for $0 \leq x \leq 1$ by storing the coefficients a_k or the particular values $f(\bar{x}_1)$ which enable us to compute the a_k.

3. APPLICATION TO DYNAMIC PROGRAMMING

Consider now the application of these ideas to the computational solution of the functional equations of dynamic programming. Suppose that we wish to compute the sequence $\{f_n(x)\}$ determined by

$$(3.1) \qquad f_N(x) = \max_{0 \leq y \leq x} [g_N(y) + f_{N-1}(x - y)],$$

$N \geq 2$, given that $f_1(x) = g_1(x)$. To avoid the tabulation of each of the functions $f_N(x)$ at the x-grid $[0,\Delta,\ldots,K\Delta]$, where K may be a large number, we approximate to each function $f_N(x)$ in the manner indicated above. Starting with $f_1(x)$, we store the values $f_1(x_i)$, $i = 1,2,\ldots,R$, needed to evaluate $f_1(x - y)$ in the formula determining $f_2(x)$,

$$(3.2) \qquad f_2(x) = \max_{0 \leq y \leq x} [g_2(x_2) + f_1(x - y)].$$

We then determine successively $f_2(x_1)$, $f_2(x_2)$,..., $f_2(x_R)$, a set of values which stores the function $f_2(x)$. This process is then repeated.

Although the calculation of $f_{n-1}(x - y)$ using (2.2) is far more time-consuming than taking a value from storage, we expect to gain time over-all because of the fact that we are required to calculate only a few values, $f_n(x_i)$, $i = 1,2,\ldots,R$, at each stage.

4. THE LEGENDRE POLYNOMIALS

Since in allocation processes we have a range $[0,x_0]$ which stays constant as the process continues from stage to stage, we can normalize and consider the basic interval to be $[0,1]$. Since the interval is finite and we want, at the moment, a polynomial approximation, we shall employ Legendre polynomials.

Let $P_k(x)$ denote the standard Legendre polynomial, defined over $[-1,1]$, and let $\varphi_k(x)$ be defined by the relation

$$\varphi_k(x) = (2k + 1)^{1/2} P_k(2x - 1). \tag{4.1}$$

The sequence $\{\varphi_k(x)\}$ is then orthonormal over $[0,1]$, i.e.,

$$\int_0^1 \varphi_k(x)\varphi_\ell(x)dx = \delta_{k\ell}. \tag{4.2}$$

From the standard recurrence relations for $P_k(x)$, we obtain the relation

$$(4.3)\qquad \varphi_1(x) = 1, \quad \varphi_2(x) = 3^{1/2}(2x-1),$$

$$\varphi_{i+2}(x) = \frac{(2i+3)^{1/2}}{(i+1)}\left[(2i+1)^{1/2}(2x-1)\varphi_{i+1}(x) - \frac{i}{(2i-1)^{1/2}}\varphi_i(x)\right].$$

This relation makes the evaluation of the sequence of values of $\varphi_i(x)$ for any x a relatively simple matter.

5. EXAMPLES

In order to experiment with this new approximating procedure, we devised a FORTRAN program for the IBM-7090 whereby at each stage, after obtaining the new coefficients $a_k^{(i)}$, we computed $f_i(x)$ from (2.2) for a succession of as many values of x as desired, and printed the result after each computation. We used two modes of output, either a list of numerical values, $[x, f_i(x)]$, or a graphical plot (done directly by the 7090) of x versus $f_i(x)$.

We experimented with several types of functions $g_i(x)$ for which the results could be derived from analytic considerations.* Using the known analytic results as a checkpoint, we varied the parameters R, M, and the grid size H, in order to determine the degree of accuracy we might expect in general. We found remarkably good agreement to 2

*Oliver Gross was of considerable assistance to us in this respect.

or more significant figures with a relatively low order of approximation, namely $R = 5$, $M = 6$. This is encouraging from the point of view of extending the method in the experimental investigations of higher-dimensional allocation processes where, as pointed out above, time and storage aspects become significant. We also incorporated in our program restraints of the form $0 \leq a_i \leq x_i \leq b_i \leq x$, since constraints of this nature often occur in applications.

a. Time Estimates

Following are some estimates of the execution time required on the 7090:

1. 10 seconds for 4 stages, $R = 5$, $M = 6$ and a grid of 0.05 both for the search and output listing.

2. 3 minutes for a total of 4 cases of 10 stages each; $R = 3$, $M = 4$; $R = 5$, $M = 6$; $R = 7$, $M = 8$; $R = 10$, $M = 11$. The grid in the search for the maximum is 0.01 and the output listing is given for every x along the grid.

3. 3 minutes for the total of 4 cases mentioned above, 10 stages per case and a grid of 0.01 for the search. The output is a graph where the independent variable x is listed at intervals of 0.025.

b. Numerical Results

1. $g_i(x) = i(x)^{1/2}$. Using the Schwarz inequality, we readily obtain the values

$$f_N(x) = \left(\frac{N}{6}(N + 1)(2N + 1)x\right)^{1/2}.$$

For $R = 10$, $M = 11$, $H = 0.01$ we found poor agreement at the origin. This is to be expected because of the infinite slope at $x = 0$. Agreement between exact and computed values was good as soon as x moved away from the singular value 0, as can be seen from the following table:

Function	Exact	Computed
$f_1(0)$	0.0	0.064
$f_1(2)$	0.448	0.447
$f_1(1)$	1.00	1.00
$f_{10}(0)$	0.0	3.13
$f_{10}(1)$	19.6	19.6

2. $g_1(x) = 1(x + 0.1)^{1/2}$. We avoided the previous difficulty at $x = 0$ (see the computed value of $f_{10}(0)$ above), but found the function still rather sensitive near the origin. As N (the number of stages) increased, the agreement at $x = 0$ decreased.

3. $g_1(x) = 1(x + 1)^{1/2}$. The Schwarz inequality yields the upper bound

$$f_N(x) \leq \left(\frac{N}{6}(N + 1)(2N + 1)(x + N)\right)^{1/2}.$$

However, since the x_1 are subject to the restraint $0 \leq x_1 \leq x$, we do not necessarily achieve the upper bound. Some exact values based on an analysis by O. A. Gross are listed as check points, $R = 10$, $M = 11$, $H = 0.01$.

Function	Upper bound	Exact value	Computed value
$f_1(0)$	1.00	1.00	1.00
$f_1(1)$	1.41	1.41	1.41
$f_3(0)$	6.48	——	6.00
$f_3(.5)$	7.00	6.67	6.67
$f_7(.35)$	32.1	29.1	29.1
$f_{10}(1)$	——	59.3	59.3

4. $g_1(x) = e^{-5/(1+10x)}$, R = 10, M = 11, H = .01.

Function	Exact	Computed
$f_2(.2)$	0.196	0.197
$f_2(.7)$	0.659	0.659
$f_3(.2)$	0.203	0.204
$f_3(.9)$	0.860	0.862
$f_5(.2)$	0.218	0.217
$f_{10}(1)$	1.001	1.001

6. TWO-DIMENSIONAL APPROXIMATION

The problem of maximizing the function

$$(6.1) \qquad \sum_{i=1}^{N} g_i(x_i, y_i)$$

over the domain

$$(6.2) \qquad \sum_{i=1}^{N} x_i = x, \quad 0 \le a_i \le x_i \le b_i, \quad x_i \le x,$$

$$\sum_{i=1}^{N} y_i = y, \quad 0 \le c_i \le y_i \le d_i, \quad y_i \le y,$$

can be readily handled by the methods described in Secs. 2 and 3.

The dynamic programming recurrence relation is:

$$f_N(x,y) = \max_R \left[g_N(x_N,y_N) + f_{N-1}(x - x_N, y - y_N) \right],$$

where R is determined by $a_N \le x_N \le \min(x,b_N)$, $c_N \le y_N \le \min(y,d_N)$. Let $\bar{x}_i$, $i = 1,\ldots,R$, be the roots of the normalized Legendre polynomial $\phi_R(x)$, and let $\{\bar{y}_j\}$ be a duplicate set of these quadrature points. Each function $f_N(x,y)$ is expressed approximately by the relation

$$f_N(x,y) = \sum_{r=1}^{M} \sum_{s=1}^{M} a_{r,s}^{(N)} \phi_r(x) \phi_s(y), \tag{6.3}$$

where the coefficients, using the quadrature method, are given by

$$a_{k,s}^{(N)} = \sum_{i=1}^{R} \sum_{j=1}^{R} w_i w_j f_N(\bar{x}_i,\bar{y}_j) \phi_k(\bar{x}_i,\bar{y}_j). \tag{6.4}$$

As in the one-dimensional case, we start with the known values $f_1(x,y) = g_1(x_1,y_1)$, $x_1 = x$, $y_1 = y$. In stage n we store the values $f_n(\bar{x}_i,\bar{y}_j)$, for $i,j = 1,2,\ldots,M$, and then compute and store the values $a_{r,s}^{(n)}$ in the storage allotted to the previous stage. The latter coefficients are utilized in the computation of $f_n(x - x_{n+1}, y - y_{n+1})$ to obtain the values of $f_{n+1}(x_i,y_j)$ in the next stage.

7. EXAMPLES

a. Time Estimate

The execution time required on the 7090 was 2 minutes for 4 stages, with $R = 5$, $M = 6$, a grid of $H = 0.05$ in the two-dimensional search, and an output listing of 3 test values of $f(x,y)$ in each stage.

b. Numerical Results

1. $g_1(x,y) = (2i - 1)^{1/2}(xy)^{1/4}$. Using the Hölder inequality, we readily obtain the exact values $f_n(x,y) = n(xy)^{1/4}$, $x_n(x,y) = (2n-1)x/n^2$, $y_n(x,y) = (2n-1)y/n^2$. Our results for $R = 5$, $M = 6$, and a grid of 0.05 in the search are:

Function	Exact	Computed
$f_2(.5,.5)$	1.40	1.40
$x_2(.5,.5)$	0.375	0.35 (to the nearest 0.05)
$f_4(1,1)$	4.00	3.91

Here again the agreement was poor at the origin, presumably because of the singular behavior of $(xy)^{1/4}$ at $x = 0$, $y = 0$. To confirm this hypothesis, we considered the next case.

2. $g_1(x,y) = (x + iy)/(1 + x + iy)$. This case, as well as the theoretical values, was suggested by O. A. Gross, and he determined the exact values.

Function	Exact	Computed
$f_2(1,1)$	1.17	1.17
$f_2(1,0)$	0.667	0.647

8. DISCUSSION

As can be seen, the agreement in general is quite satisfactory. We can obtain reasonably accurate values of the maximum return and of the optimal allocation policy using small amounts of machine time.

Combining these techniques with the method of the Lagrange multiplier, we can expect to solve three- and four-dimensional resource allocation problems. Extending the method to cover the approximation of functions of 3, 4, 5 and 6 variables, we can treat Hitchcock-Koopmans allocation processes of quite high dimension.

Finally, if we combine these techniques—polynomial approximations and Lagrange multipliers—with that of successive approximations, there should be very few allocation processes which still resist our efforts.

Appendix A

```
C  1 DIMENSIONAL ALLOCATION VIA DYNAMIC PROGRAMMING,APPROXIMATIONS
*       LIST
*       LABEL
C649501
       COMMON N,M,JR,H,IH,L,Y,XL,W,A,P,G,NI,A1,B1
      X ,IH1,L1,L2,H1,S,S1,HOLAST,HOLBL,HOLDOT
       DIMENSION XL(20),W(20),A(20),P(20),F(20),X(20),A1(10),B1(10)
     1 CALL INPUT
       IF(N-100)2,3,3
C      STAGE 1
     2 NI =1
       DO 4 K=1,M
       SUM=0.
       DO 5 J=1,JR
       Y=XL(J)
       CALL RETG
       CALL POLY
       F(J)=G
       X(J)=Y
     5 SUM=SUM+W(J)*F(J)*P(K)
       A(K)=SUM
     4 CONTINUE
       OUTPUT 42,60,NI,JR,M
    60 FORMAT(20H1   ITERATIVE STAGE   I5,5H   R=I3,5H   M=I3)
       OUTPUT 42,661
   661 FORMAT(53HO      DECISION            RETURN  AT R ROOT VALUES      )
       OUTPUT 42,61,(X(J),F(J),J=1,JR)
    61 FORMAT(1H 2E20.8)
       OUTPUT 42,550
   550 FORMAT (30HO   COEFFICIENTS  A(K)                                  )
       OUTPUT 42,50,(A(K),K=1,M)
    50 FORMAT(1H E20.8)
       CALL OUT
C      STAGE NI GREATER THAN 1
       DO 6 N2=2,N
       NI=N2
       DO 7 J=1,JR
       Y=0.
       CALL RETG
       Y=XL(J)-0.
       CALL POLY
       SUM=0.
       DO 8 K=1,M
     8 SUM=SUM+A(K)*P(K)
       F(J)=G+SUM
       X(J)=0.
       DO 9 I=2,IH
       XI=I
       IF((XI-1.)*H-A1(N2))9,110,110
   110 CONTINUE
       IF((XI-1.)*H-B1(N2))111,111,11
   111 CONTINUE
       IF((XI-1.)*H-XL(J))10,10,11
    10 Y=(XI-1.)*H
       CALL RETG
       Y=XL(J)-(XI-1.)*H
       CALL POLY
       SUM=0.
       DO 12 K=1,M
    12 SUM=SUM+A(K)*P(K)
```

```
      PH=G+SUM
      IF(PH-F(J))9,9,15
   15 F(J)=PH
      X(J)=(XI-1.)*H
    9 CONTINUE
   11 CONTINUE
    7 CONTINUE
      OUTPUT 42,60,NI,JR,M
      OUTPUT 42,661
      OUTPUT 42,61,(X(J),F(J),J=1,JR)
      DO 16 K=1,M
      SUM=0.
      DO 17 J=1,JR
      Y=XL(J)
      CALL POLY
   17 SUM=SUM+W(J)*F(J)*P(K)
      A(K)=SUM
   16 CONTINUE
      OUTPUT 42,550
      OUTPUT 42,50,(A(K),K=1,M)
      CALL OUT
    6 CONTINUE
      GO TO 1
    3 CALL EXIT
      STOP
      END
*      LIST
*      LABEL
C649502
C     INPUT ALL CONSTANTS
      SUBROUTINE INPUT
      COMMON N,M,JR,H,IH,L,Y,XL,W,A,P,G,NI,A1,B1
     X ,IH1,L1,L2,H1,S,S1,HOLAST,HOLBL,HOLDOT
      DIMENSION XL(20),W(20),A(20),P(20),F(20),X(20),A1(10),B1(10)
      INPUT 41,20,N,M,JR,IH,L
      INPUT 41,21,(XL(J),J=1,JR),(W(I),I=1,JR),H,(A1(I),I=1,N),
     X   (B1(J),J=1,N)
      OUTPUT 42,22,N,M,JR,IH,L
      OUTPUT 42,23,(XL(J),J=1,JR),(W(I),I=1,JR),C,H,(A1(I),I=1,N),
     X   (B1(J),J=1,N)
      INPUT 41,1001,IH1,L1,L2
      INPUT 41,1002,H1,S,S1
      INPUT 41,1000,HOLAST,HOLBL,HOLDOT
      RETURN
 1000 FORMAT(3A1)
 1001 FORMAT(I10)
 1002 FORMAT(F10.3)
   20 FORMAT(5I10)
   21 FORMAT(E20.8)
   22 FORMAT(1H15I10)
   23 FORMAT(1H E20.8)
      END
*      LIST
*      LABEL
C649503
C     COMPUTE RETURN G AS FUNCTION OF Y
      SUBROUTINE RETG
      COMMON N,M,JR,H,IH,L,Y,XL,W,A,P,G,NI,A1,B1
     X ,IH1,L1,L2,H1,S,S1,HOLAST,HOLBL,HOLDOT
      DIMENSION XL(20),W(20),A(20),P(20),F(20),X(20),A1(10),B1(10)
```

```
      XNI=NI
      G=XNI*SQRTF(Y+1.)
      RETURN
      END
*      LIST
*      LABEL
C649504
C     NORMALIZED POLYNOMIALS P(K) AS FUNCTIONS OF Y
      SUBROUTINE POLY
      COMMON N,M,JR,H,IH,L,Y,XL,W,A,P,G,NI,A1,B1
     X ,IH1,L1,L2,H1,S,S1,HOLAST,HOLBL,HOLDOT
      DIMENSION XL(20),W(20),A(20),P(20),F(20),X(20),A1(10),B1(10)
      P(1)=1.
      P(2)=SQRTF(3.)*(2.*Y-1.)
      M1=M-2
      DO 40 I=1,M1
      BI=I
      D=SQRTF(2.*BI+3.)/(BI+1.)
      E=SQRTF(2.*BI+1.)
      B=BI/SQRTF(2.*BI-1.)
   40 P(I+2)=D*(E*(2.*Y-1.)*P(I+1)-B*P(I))
      RETURN
      END
*      LIST
*      LABEL
C649505
C     COMPUTE AND OUTPUT TOTAL RETURN FUNCTION AT STAGE NI
      SUBROUTINE OUT
      COMMON N,M,JR,H,IH,L,Y,XL,W,A,P,G,NI,A1,B1
     X ,IH1,L1,L2,H1,S,S1,HOLAST,HOLBL,HOLDOT
      DIMENSION XL(20),W(20),A(20),P(20),F(20),X(20),A1(10),B1(10)
      DIMENSION Z(110)
      OUTPUT 42,75,NI,JR,M
   75 FORMAT(20H1   ITERATIVE STAGE   I5,5H   R=I3,5H   M=I3)
      IF(L-1)13,18,18
   13 OUTPUT 42,774
  774 FORMAT(40HO    RESOURCE                RETURN                )
      DO 72 I9=1,IH
      XI=I9
      Y=(XI-1.)*H
      CALL POLY
      SUM=0.
      DO 73 K=1,M
   73 SUM=SUM+A(K)*P(K)
      FF=SUM
      OUTPUT 42,74,Y,FF
   74 FORMAT(1H F15.3,E20.8)
   72 CONTINUE
      GO TO 71
   18 CONTINUE
       DO 772 I9=1,IH1
      XI=I9
      Y=(XI-1.)*H1
      CALL POLY
      SUM=0.
      DO 773 K=1,M
  773 SUM=SUM+A(K)*P(K)
      FF=SUM
      N5=FF*S+S1
      DO 1003 I5=1,110
```

```
1003 Z(I5)=HOLBL
     DO 1004 L5=1,L2
     IF(I9-L1*L5-1)76,77,76
  77 DO 78 I6=1,110
     Z(I6)=HOLDOT
  78 CONTINUE
  76 CONTINUE
1004 CONTINUE
     DO 1005 I5=1,110,10
1005 Z(I5)=HOLDOT
     Z(N5)=HOLAST
     OUTPUT 42,1006,Y,(Z(I7),I7=1,105)
 772 CONTINUE
  71 CONTINUE
     RETURN
1006 FORMAT(1H F10.3,109A1)
     END
```

Appendix B

```
C       2 DIMENSIONAL ALLOCATION VIA DYNAMIC PROGRAMMING,APPROXIMATIONS
*       LIST
*       LABEL
C649500
        COMMON N,M,JR,NI,X,Y,G,F1,W,V,XL,YL,A,P,PH,F,B,C,IH,H,A1,B1,C1,D1
        DIMENSION W(20),V(20),XL(20),YL(20),A(20,20),P(20),PH(20),F(20,20)
       X     ,B(3),C(3),SUM(20),SUM2(20),PX(20,20),PY(20,20)
       X   ,B1(20),A1(20),C1(20),D1(20)
      1 CALL INPUT
        IF(N-100)2,3,3
C       STAGE 1
      2 NI=1
        OUTPUT 42,60,NI,JR,M
     60 FORMAT(20H1    ITERATIVE STAGE    I5,5H    R=I3,5H    M=I3)
        OUTPUT 42,54
     54 FORMAT(90H0            X                        Y                        F
       X          XDEC       YDEC                                                )
        DO 4 I=1,JR
        X=XL(I)
        CALL POLY
        DO 600 K=1,M
    600 PX(K,I)=P(K)
        DO 41 J=1,JR
        Y=YL(J)
        CALL PHOLY
        DO 601 L=1,M
    601 PY(L,J)=PH(L)
        CALL RETG
        F(I,J)=G
        XDEC=XL(I)
        YDEC=YL(J)
        OUTPUT 42,61,X,Y,F(I,J),XDEC,YDEC
     61 FORMAT(1H 3E20.6,2F10.3)
     41 CONTINUE
      4 CONTINUE
        DO 5 K=1,M
        DO 51 L=1,M
        DO 52 I=1,JR
        SUM(I)=0.
        DO 402 J=1,JR
    402 SUM(I)=SUM(I)+V(J)*F(I,J)*PY(L,J)
     52 CONTINUE
        SUM1=0.
        DO 70 I=1,JR
     70 SUM1=SUM1+W(I)*PX(K,I)*SUM(I)
        A(K,L)=SUM1
     51 CONTINUE
      5 CONTINUE
        OUTPUT 42,50
     50 FORMAT(30H1     COEFFICIENTS A(K,L)                )
        OUTPUT 42,53,((A(I,J),I=1,M),J=1,M)
     53 FORMAT(1H 20.6)
        OUTPUT 42,556
        DO 55 I=1,3
        X=B(I)
```

```
      Y=C(I)
      CALL OUT
   55 CONTINUE
C     STAGE NI GREATER THAN 1
      DO 6 N2=2,N
      NI=NI+1
      OUTPUT 42,60,NI,JR,M
      OUTPUT 42,54
      DO 8 I=1,JR
      DO 80 J=1,JR
      X=0.
      Y=0.
      CALL RETG
      X=XL(I)-0.
      Y=YL(J)-0.
      CALL TOTRET
      F(I,J)=F1
      XDEC=0.
      YDEC=0.
      DO 9 I1=1,IH
      XI=I1
      IF((XI-1.)*H-A1(I))9,99,99
   99 CONTINUE
      IF((XI-1.)*H-B1(I))991,991,11
  991 CONTINUE
      IF((XI-1.)*H-XL(I))10,10,11
   10 CONTINUE
      DO 91 J1=1,IH
      XJ=J1
      IF((XJ-1.)*H-C1(J))91,992,992
  992 CONTINUE
      IF((XJ-1.)*H-D1(J))993,993,9
  993 CONTINUE
      IF((XJ-1.)*H-YL(J))100,100,9
  100 X=(XI-1.)*H
      Y=(XJ-1.)*H
      CALL RETG
      X=XL(I)-(XI-1.)*H
      Y=YL(J)-(XJ-1.)*H
      CALL TOTRET
      F2=F1
      IF(F2-F(I,J))9?,91,15
   15 F(I,J)=F2
      XDEC=(XI-1.)*H
      YDEC=(XJ-1.)*H
   91 CONTINUE
    9 CONTINUE
   11 CONTINUE
      OUTPUT 42,61,  XL(I),YL(J),F(I,J),XDEC,YDEC
   80 CONTINUE
    8 CONTINUE
      DO 400 K=1,M
      DO 401 L=1,M
      DO 502 I=1,JR
```

```
      SUM(I)=0.
      DO 500 J=1,JR
  500 SUM(I)=SUM(I)+V(J)*F(I,J)*PY(L,J)
  502 CONTINUE
      SUM1=0.
      DO 501 I=1,JR
  501 SUM1=SUM1+W(I)*SUM(I)*PX(K,I)
      A(K,L)=SUM1
  401 CONTINUE
  400 CONTINUE
      OUTPUT 42,50
      OUTPUT 42,53,((A(I,J),I=1,M),J=1,M)
      OUTPUT 42,556
  556 FORMAT(54H0  TEST     X                    Y                    F    )
      DO 555 I=1,3
      X=B(I)
      Y=C(I)
      CALL OUT
  555 CONTINUE
    6 CONTINUE
      GO TO 1
    3 CALL EXIT
      STOP
      END
*      LIST
*      LABEL
C649501
C     INPUT ALL CONSTANTS
      SUBROUTINE INPUT
      COMMON N,M,JR,NI,X,Y,G,F1,W,V,XL,YL,A,P,PH,F,B,C,IH,H,A1,B1,C1,D1
      DIMENSION W(20),V(20),XL(20),YL(20),A(20,20),P(20),PH(20),F(20,20)
     X    ,B(3),C(3),SUM(20),SUM2(20),PX(20,20),PY(20,20)
     X  ,B1(20),A1(20),C1(20),D1(20)
      INPUT 41,20,N,M,JR,IH
      INPUT 41,21,(XL(J),J=1,JR),(W(I),I=1,JR),(YL(J),J=1,JR),
     X            (V(I),I=1,JR),H,(B(I),I=1,3),(C(J),J=1,3)
      OUTPUT42,22,N,M,JR,IH
      OUTPUT42,23,(XL(J),J=1,JR),(W(I),I=1,JR),(B(I),I=1,3),H
      INPUT 41,233,(A1(I),I=1,JR),(B1(I),I=1,JR),(C1(I),I=1,JR),
     X   (D1(I),I=1,JR)
  233 FORMAT(F10.3)
   20 FORMAT(4I10)
   21 FORMAT(E20.8)
   22 FORMAT(1H14I10)
   23 FORMAT(1H E20.8)
      RETURN
      END
*      LIST
*      LABEL
C649502
C     COMPUTE RETURN G AS FUNCTION OF X AND Y
      SUBROUTINE RETG
      COMMON N,M,JR,NI,X,Y,G,F1,W,V,XL,YL,A,P,PH,F,B,C,IH,H,A1,B1,C1,D1
      DIMENSION W(20),V(20),XL(20),YL(20),A(20,20),P(20),PH(20),F(20,20)
```

```
      X     ,B(3),C(3),SUM(20),SUM2(20),PX(20,20),PY(20,20)
      X   ,B1(20),A1(20),C1(20),D1(20)
       XNI=NI
       G=SQRTF(2.*XNI-1.)*SQRTF(SQRTF((X+.1)*(Y+.1)))
       RETURN
       END
*       LIST
*       LABEL
C649503
C      NORMALIZED POLYNOMIALS P(K)AS FUNCTIONS OF X
       SUBROUTINE POLY
       COMMON N,M,JR,NI,X,Y,G,F1,W,V,XL,YL,A,P,PH,F,B,C,IH,H,A1,B1,C1,D1
       DIMENSION W(20),V(20),XL(20),YL(20),A(20,20),P(20),PH(20),F(20,20)
      X     ,B(3),C(3),SUM(20),SUM2(20),PX(20,20),PY(20,20)
      X   ,B1(20),A1(20),C1(20),D1(20)
       P(1)=1.
       P(2)=SQRTF(3.)*(2.*X-1.)
       M1=M-2
       DO 40 I1=1,M1
       BI=I1
       D=SQRTF(2.*BI+3.)/(BI+1.)
       E=SQRTF(2.*BI+1.)
       T=BI/SQRTF(2.*BI-1.)
   40 P(I1+2)=D*(E*(2.*X-1.)*P(I1+1)-T*P(I1))
       RETURN
       END
*       LIST
*       LABEL
C649504
C      NORMALIZED POLYNOMIALS PH(K) AS FUNCTIONS OF Y
       SUBROUTINE PHOLY
       COMMON N,M,JR,NI,X,Y,G,F1,W,V,XL,YL,A,P,PH,F,B,C,IH,H,A1,B1,C1,D1
       DIMENSION W(20),V(20),XL(20),YL(20),A(20,20),P(20),PH(20),F(20,20)
      X     ,B(3),C(3),SUM(20),SUM2(20),PX(20,20),PY(20,20)
      X   ,B1(20),A1(20),C1(20),D1(20)
       PH(1)=1.
       PH(2)=SQRTF(3.)*(2.*Y-1.)
       M1=M-2
       DO 90 I1=1,M1
       BI=I1
       D=SQRTF(2.*BI+3.)/(BI+1.)
       E=SQRTF(2.*BI+1.)
       T=BI/SQRTF(2.*BI-1.)
   90 PH(I1+2)=D*(E*(2.*Y-1.)*PH(I1+1)-T*PH(I1))
       RETURN
       END
*       LIST
*       LABEL
C649505
C      COMPUTE TOTAL RETURN AS FUNCTION OF X,Y
       SUBROUTINE TOTRET
       COMMON N,M,JR,NI,X,Y,G,F1,W,V,XL,YL,A,P,PH,F,B,C,IH,H,A1,B1,C1,D1
       DIMENSION W(20),V(20),XL(20),YL(20),A(20,20),P(20),PH(20),F(20,20)
      X     ,B(3),C(3),SUM(20),SUM2(20),PX(20,20),PY(20,20)
```

```
     X   ,B1(20),A1(20),C1(20),D1(20)
      CALL POLY
      CALL PHOLY
      DO 81 IL=1,M
      SUM2(IL)=0.
      DO 83 IK=1,M
   83 SUM2(IL)=SUM2(IL)+P(IK)*A(IK,IL)
   81 CONTINUE
      SUM3=0.
      DO 84 IL=1,M
   84 SUM3=SUM3+PH(IL)*SUM2(IL)
      F1=G+SUM3
      RETURN
      END
*      LIST
*      LABEL
C649506
C     TOTAL RETURN FROM POLYNOMIAL APPROXIMATION,FUNCTION OF X,Y
      SUBROUTINE OUT
      COMMON N,M,JR,NI,X,Y,G,F1,W,V,XL,YL,A,P,PH,F,B,C,IH,H,A1,B1,C1,D1
      DIMENSION W(20),V(20),XL(20),YL(20),A(20,20),P(20),PH(20),F(20,20)
     X     ,B(3),C(3),SUM(20),SUM2(20),PX(20,20),PY(20,20)
     X   ,B1(20),A1(20),C1(20),D1(20)
      CALL POLY
      CALL PHOLY
      DO 281 IL=1,M
      SUM2(IL)=0.
      DO 283 IK=1,M
  283 SUM2(IL)=SUM2(IL)+P(IK)*A(IK,IL)
  281 CONTINUE
      SUM3=0.
      DO 284 IL=1,M
  284 SUM3=SUM3+PH(IL)*SUM2(IL)
      FF=SUM3
      OUTPUT 42,261,X,Y,FF
      RETURN
  261 FORMAT(1H 3E20.6)
      END                                                    000255

                                                             000255
```

REFERENCES

1. Bellman, R., and S. Dreyfus, Applied Dynamic Programming, The RAND Corporation, Report R-352, to appear; also to be published by Princeton University Press, Princeton, New Jersey.

2. Bellman, R., R. Kalaba, and M. Prestrud, "On a New Computational Solution of Time-dependent Transport Processes--I: One-dimensional Case," Proc. Nat. Acad. Sci. USA, Vol. 47, 1961, pp. 1072-1074.